UNITED STATES PATENT APPLICATION FOR

EXTENDING ROLE SCOPE IN A DIRECTORY SERVER SYSTEM

Inventors:

KARINE EXCOFFIER

ROBERT BYRNE

EXTENDING ROLE SCOPE IN A DIRECTORY SERVER SYSTEM

RELATED APPLICATION

This Application claims priority to French Patent Application, Number 0208491,

filed on July 5, 2002, in the name of SUN Microsystems, Inc., which application is

5   hereby incorporated by reference.

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

15   The present invention relates to the field of distributed computer systems.

Specifically, embodiments of the present invention relate extending the scope of

a role in a directory server system.

BACKGROUND ART

20   In certain fields of technology, complete computer systems, including a diversity

of equipment, are optimized for storing and retrieving data. Such systems may

provide services to user machines related to a local network, e.g., an Intranet, or to a global network, e.g., the Web network.

It is desirable that network users can access, upon a query, a large number of

5   data, making it possible for the network users to create their own dynamic web site or to consult a dynamic web site, for example an e-commerce site on a multi platform computer system (Solaris™, Windows NT™). These queries are directed to a directory, e.g., an LDAP directory, and managed by a directory server. It is further desirable that this access to a large number of data be made possible

10   more rapidly for each query arriving after a first query.

SUMMARY OF THE INVENTION

The present invention provides a method and system for extending role scope in a directory server system. In one embodiment, a directory server system comprises a directory server and a role mechanism. The directory server

5   interacts with entries organized in a tree structure. The entries comprise user entries and role entries. The role entries define a role and have an associated scope in the tree structure. The scope is defined from the location of the role entries in the tree structure, according to a predefined rule. The role mechanism is capable of attaching a role of an existing role entry to a user entry subject to a

10   first condition. The first condition comprises a role membership condition and the user entry belonging to the scope of the existing role entry. The role mechanism is further capable of determining whether the existing role entry has extra data designating an extra scope, and, if so, of attaching a role of the existing role entry to a user entry subject to a second condition. The second condition comprises

15   the role membership condition and the user entry belonging to the extra scope of the existing role entry.

Another embodiment is a method of operating a directory server system. In the method, the directory server interacts with entries organized in a tree structure, in

20   which the entries comprise user entries and role entries. Each role entry defines a role and has an associated scope in the tree. The scope is defined from the location of the role entry in the tree, according to a predefined rule. The role of an

existing role entry is attached to a user entry subject to a first condition, which comprises a role membership condition and the fact that the user entry belongs to the scope of the existing role entry. The method comprises adding extra role data to the existing role entry, the extra data identifying an extra scope in the tree

5   for the existing role entry. The method further comprises attaching the role of the existing role entry to a user entry subject to a second condition comprising the role membership condition and the user entry belonging to the extra scope of the existing role entry.

10   Embodiments of the present invention provide these advantages and others not specifically mentioned above but described in the sections to follow.

BR1EF DESCR1PTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and form a part of this specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the invention:

5

Figure 1 is a general diagram of a computer system in which embodiments of the invention are applicable.

Figure 2 illustrates a typical LDAP exchange between an LDAP client and an

10   LDAP server, and between the LDAP server and additional servers.

Figure 3 represents the general structure of an LDAP directory.

Figure 4 shows an exemplary portion of an LDAP tree.

15

Figure 5 represents attribute types and values of an entry.

Figure 6 is a portion of an LDAP tree showing the scope of a role, according to the prior art.

20

Figure 7 is a table showing three types of LDAP roles according to the prior art.

Figures 8a to 8c represent the structure of the three types of roles shown in Figure 7.

Figure 9 shows a portion of an LDAP tree structure, according to the prior art.

5

Figure 10 is a general flowchart for determining whether a given entry is a member of an existing role, in accordance with an embodiment of this invention.

Figure 11 is a table, showing syntax of an extended role, in accordance with an

10   embodiment of the invention.

Figure 12 is a flowchart for determining whether a given entry is a member of an existing role, in accordance with an alternative embodiment of this invention.

15   Figure 13a is a flowchart for enumerating the roles of a given entry according to the prior art.

Figure 13b is a flowchart for enumerating the roles of a given entry according to an embodiment of the present invention.

20

Figure 14 represents portions of an LDAP tree comprising extending roles in accordance with an embodiment of the present invention.

Figure 15a, Figure 15b, Figure 15c, Figure16, Figure 17, Figure 18a, Figure 18b, and Figure 18c represent the structure of role caches associated with top suffixes according to embodiments of the invention.

5

Figure 19 shows a tree structure portion having an extending role.

DETAILED DESCR1PTION OF THE INVENTION

In the following detailed description of the present invention, extending role scope in a directory server system, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it

5   will be recognized by one skilled in the art that the present invention may be practiced without these specific details or with equivalents thereof. In other instances, well-known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the present invention.

10

Embodiments of the present invention may be defined as an apparatus or system, and/or as software code for implementing the method, or for use in the system, and/or as portions of such software code, in all their alternative embodiments to be described hereinafter.

15

Making reference to software entities imposes certain conventions in notation. Particularly, an expression indicated between the quote signs " " may be used to design LDIF extracts and an expression in *italics* may be used for representing an attribute and an object class.

20

As they may be cited in this specification, Sun™, Sun Microsystems™ and Sun One™ are trademarks of Sun Microsystems, Inc.

This invention may be implemented in a computer system, or in a network comprising computer systems. Figure 1 represents an example of the hardware of such computer systems. The hardware comprises:

5

- a processor CPU 11, e.g., an Ultra-Sparc;

- a program memory 12, e.g., an EPROM, a RAM, or Flash memory;

- a working memory 13, e.g., a RAM of any suitable technology;

- a mass memory 14, e.g., one or more hard disks;

10   - a display 15, e.g., a monitor;

- a user input device 16, e.g., a keyboard and/or a mouse;

- a network interface device 21 connected to a communication medium 20, which is in communication with other computers. Network interface device 21 may be of the type of Ethernet, or of the type of ATM. Medium 20 may be based on wire

15   cables, fiber optics, or radio-communications, for example.

Data may be exchanged between the components of Figure 1 through a bus system 10, represented as a single bus for simplification of the drawing. Bus systems may include a processor bus, e.g., PCI, connected via appropriate

20   bridges to, e.g., an ISA or an SCSI bus.

The data exchanged are handled by a resource provider using a server to deliver data to user computers, or to store the data provided by the user computers. Browsers, e.g., Internet Explorer™, are further provided on user computers, to enable users, to make requests to retrieve or store data. The resource provider

5   makes it possible for user computers on a network to share data of any kind.

iPlanet E-commerce Solutions, now Sun One™ E-commerce Solutions, has developed a "net-enabling" platform called the Internet Service Deployment Platform (ISDP). ISDP includes multiple, integrated layers of software that

10   provide a full set of services supporting application development, e.g., business-to-business exchanges, communications and entertainment vehicles, and retail Web sites.

Sun One™ Directory Server provides a centralized directory service directory

15   service for an intranet or an extranet. A directory service represents a collection of software, hardware, and processes that are able to deliver and store information. The directory service generally includes one or more directory client programs that can access the data stored in the directory, e.g., names, telephone numbers, or addresses.

20

The Sun One™ Directory Server is a general-purpose directory that stores all information in a single, network-accessible repository. The Sun One™ Directory

Server provides the standard protocol LDAP and an application programming

interface (API) to access the information contained by the Sun One™ Directory

Server.

5   LDAP is the Internet standard for directory lookups, just as the Simple Mail

Transfer Protocol (SMTP) is the Internet Standard for delivering e-mail and the

Hypertext Transfer Protocol (HTTP) is the Internet standard for delivering

documents. Technically, LDAP is defined as on-the-wire bit protocol (similar to

HTTP) that runs over Transmission Control Protocol/ Internet Protocol (TCP/IP).

10   It specifies the interaction between clients and servers and determines how

LDAP queries and responses are carried over the IP network.

An LDAP-compliant directory, such as the Sun One™ Directory Server, leverages

a single, master directory that contains all users, groups and access information.

15   The directory is hierarchical, not relational and is particularly fitted for reading

while offering a high reliability and a high scalability.

For example, the directory can be used to provide information technology

managers with a list of all the hardware and software assets in a widely spanning

20   enterprise. Most importantly, a directory server provides resources that all

applications can use, and aids in the integration of these applications that have

previously functioned as stand-alone systems. Instead of creating an account for

each user in each system the user needs to access, a single directory entry is created for the user in the LDAP directory.

Referring now to Figure 2, LDAP defines a communication 1 between a server 17
5   and a client 18. LDAP also defines a communication 2 between LDAP server 17 and servers 17.1 to 17.n, which makes it possible for the server LDAP 17 to exchange its content (replication service) with servers 17.1 to 17.n or to access the directory of one of the servers 17.1 to 17.n (referral service) and vice versa.

10   The LDAP protocol is a message-oriented protocol. The client 18 constructs an LDAP message containing a request and sends the message to the server 17. The server 17 processes the request and sends a result, or results, back to the client 18 as a series of LDAP messages.

15   Such a client-server communication additionally lies on a specific architecture. LDAP creates a standard defining the way data are exchanged between the client computer and the directory server and defining the way data are modeled. More specifically, LDAP relies on four basic models:

20   - an information model;
          - a naming model;
          - a functional model; and
          - a security model.

The LDAP information model defines the kind of data that can be stored in a

directory. LDAP directory is populated with entries. An entry corresponds to real-

world objects, such as a person, a printer, or configuration parameters.

5

Figure 3 illustrates the general structure of an LDAP directory : the directory

server 30 executes implemented functions based on the entries 31 stored in

databases. The entries comprise configuration entries 310, user entries 311 and

administrative entries 312. These entries further interact with the schema 32

10 which is described below.

The configuration entries are stored under a subtree "cn=config". The user

entries comprise data related to the users of the directory server. Administrative

entries relate to user management and are generally implemented as LDAP

15 subentries.

An entry contains a set of attributes associated with values (e.g., common name

cn or surname sn). Each entry is uniquely identified by a distinguished name DN.

This distinguished name may be stored in the attribute dn (distinguishedName) of

20 each entry.

LDAP entries are organized in a hierarchical tree structure, called the Directory

Information Tree (DIT). Each node of the tree comprises an entry. Figure 4

illustrates an organization entry (90) with the attribute type of domain component

dc, an organizational unit entry (92) with the attribute type of organizational unit

5   *ou*, a server application entry (94) with the attribute type of common name *cn*,

and a person entry (96) with the attribute type of user ID*uid*. The entries are

connected by the directory. Each server has a particular entry called root

directory specific entry (rootDSE) which contains the description of the tree and

of its content.

10

AN LDAP Data Interchange Format (LDIF) is an ASCII text file format used to

describe directory entries and operations on those entries. It enables to create,

modify, and delete Directory entries and to import and export data among LDAP

directories. Figure 5 is a LDIF representation of an entry 404, showing the

15   attribute types 400 and their values 402.

The information model is extensible, which means that new types of information

can be added to an LDAP directory.

20   Descriptive information is stored in the attributes of the entry. Each attribute

describes a specific type of information. Attributes may have constraints that limit

the type and length of data placed in attribute values.

All entries require the *objectclass* attribute which lists the object classes to which an entry belongs. An entry can belong to one or more object classes and must satisfy all of them. The *objectclass* attribute defines which attributes are required

5   and which attributes are allowed in the entry.

For example, in Figure 5, the entry (404) represented in LDIF belongs to the object classes *top, person, organizationalPerson* and *inetOrgPerson.*

10   Each attribute has a corresponding syntax definition. The syntax definition describes the type of information provided by the attribute. The object classes, the required and allowed attributes, and the syntax definition of the attributes are listed in the directory schema.

15   The LDAP directory comprises a structure 32, represented in Figure 3, that defines object classes and attributes, and may be viewed as metadata. This structure, called the schema 32, sets the rules defining what information can be stored in the LDAP directory and how information is organized. The schema 32 specifies the required and allowed attributes that are used to store information

20   and their syntax definition. A schema checking function may be activated, thus causing the directory server to check new entries to verify whether:

- object classes and attributes attached to new entries are defined in the

schema 32,

- the attributes required for an object class according to the schema 32,

are contained in an entry attached to that object class,

5   - only attributes allowed by the object class according to the schema 32,

are contained in an entry attached to that object class.

The LDAP naming model specifies that directory entries must be hierarchical and

organized in an inverted tree structure. As mentioned above, each entry has a

10   unique name called a distinguished name *dn*. The *dn* consists of a list of the

names of all the parent entries in the directory back to the top of the directory

hierarchy, the name of the entry being at the extreme left, e.g.,

"uid=Joe,ou=people,dc=france,dc=sun,dc=com", in Figure 5. The root of the

entry is at the extreme right of the *dn*. The name at the extreme left of the *dn*,

15   "uid=Joe" in the example, is the relative distinguished name or *rdn*. Within a set

of entries sharing the same parents, the *rdn* must be unique. This ensures that

two entries in the directory tree cannot have the same *dn*.

The LDAP functional model comprises eight basic functional operations

20   (indicated thereinafter between the quote signs " ") that a user from a client

computer can perform on the directory data of an LDAP directory server:

- "bind" and "unbind": begin and end the exchange of information between LDAP

clients and the directory server;

- "add", "delete", and "modify": apply on specific entries in the DIT,

- "compare": applies on two entries to compare their content according to criteria,

5   - "search": locates specific entries in the DIT,

- "rnodifyRDN": applies to change the distinguished name *dn* of an entry.

In addition to the eight basic functional operations, the LDAP protocol defines a

framework for adding new operations to the protocol via LDAP extended

10   operations. Extended operations allow the protocol to be extended in an orderly

manner to meet new marketplace needs as they emerge.

According to another aspect of LDAP directories, entry grouping mechanisms are

provided to simplify the management of LDAP users. Roles, introduced in version

15   5 of iDS, constitute an LDAP grouping mechanism.

A role may have members, which are the entries said to possess the role. The

role mechanism enables the following operations:

- enumerating the members of a given role,

20   - determining whether a given entry possesses a particular role,

- enumerating all the roles possessed by a given entry,

Additionally, it is possible to assign a particular role to a given entry and to revoke a particular role from a given entry.

Every role is defined by its own definition entry. A role is uniquely identified by
5   the distinguished name of its definition entry. Role definition entries are LDAP subentries and therefore inherit the subentry mechanism, defined in the ISO/IEC X.509 standard, for scope. The scope of a role corresponds to the subtree of the role parent entry as illustrated by Figure 6. User entries E01 and E02 are in the scope S1 of the role R1 while user entry E11 is out of the scope of the role R1.
10   Thus, user entries E01 and E02 are likely to be members of role R1 while user entry E11 cannot be a member of role R1.

Referring to Figure 7, a role can be of "managed" type 701, "filtered" type 702 or "nested" type 703. Each type of role further has two specific object classes 71
15   that inherit from the *nsRoleDefinition* object class, and is related to specific attributes 72 (*nsRoleDN, nsRoleFilter*).

On creating a role, members may be assigned to the role as follows:
        - members of a managed role have the *nsRoleDN* attribute in their entry,
20   - members of a filtered role are entries that match the filter specified in the *nsRoleFilter* attribute of the role definition entry,

- members of a nested role are members of the roles specified in the

nsRoleDN attributes of the nested role definition entry.

Managed roles belong to the *nsSimpleRoleDefinition* object class and are thus

5 simple roles. Filtered and nested roles belong to the *nsComplexRoleDefinition*

object class are thus complex roles.

Tables I-III respectively represent a managed role, a filtered role and a nested

role, in LDIF, according to the prior art.

10

TABLE I

dn: cn = Marketing, ou = people, dc = example, dc = com
objectclass: top
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsSimpleRoleDefinition
objectclass: nsComplexRoleDefinition
cn: Marketing
description: managed role for marketing staff

TABLE II

dn : cn = SalesFilter, ou = people, dc = example, dc = com
objectclass : top
objectclass : LDAPsubentry
objectclass : nsRoleDefinition

```
objectclass : nsComplexRoleDefinition

objectclass : nsFilteredRoleDefinition

cn : SalesFilter

leFilter : description=marketing guy

description : filtered role for sales staff
```

TABLE III

```
dn : cn = MarketingSales, ou = people, dc = example, dc = com

objectclass : top

objectclass: LDAPsubentry

objectclass: nsRoleDefinition

objectclass : nsComplexRoleDefinition

objectclass : nsNestedRoleDefinition

cn : MarketingSales

nsRoleDN : cn = Marketing, ou = people, dc = example, dc = com

nsRoleDN : cn = SalesFilter, ou = people, dc = example, dc = com

description: nested role for marketing and sales staff
```

Table IV represents a user entry member of the managed role of Table I. Table V

5   rep[resents a user entry member of the filtered role of Table II.

TABLE IV

```
dn: cn = Joe, ou = people, dc = example, dc = com

objectclass : person

cn : Joe

sn : Bradford

userpassword: joepasswd

nsRoleDN : cn = Marketing, ou = people, dc = example, dc = com
```

TABLE V

dn : cn = Richard, ou = people, dc = example, dc = com
objectclass : person
cn : Richard
sn : Parker
userpassword: richardpasswd
description: marketing guy

Figures 8a, 8b and 8c respectively illustrate a managed role, filtered or nested

5   role. As represented in Figure 8a, *nsRoleDN* relates a user Entry E1 to a

managed role (*ManagedRole*), thus indicating that the entry is a member of that

role. Figure 8b shows that *nsRoleFilter* relates a filtered role (*FilteredRole*) to a

user entry E2, thus indicating that entry E2 is a member of that role. *nsRoleDN*

can also be a pointer from a nested role to another role (*ManagedRole,*

10   *FilteredRole*) as shown in Figure 8c.

LDAP servers use a computed attribute called *nsrole* to determine the roles

possessed by a given entry. The *nsrole* attribute is a computed attribute and, as

a result, it is not stored in the entry itself. This attribute is computed for each of

15   the three types of role operations identified above.

In Figure 9, a user 134 identified by the RDN "cn = rob" in the engineering

division 131 ("o=eng") is a member of the engineering role 133 ("cn = engrole").

He has the permissions associated with that role. However, he may need to access a resource, e.g., an application, requiring the permissions associated with sales role 135 ("cn = salesrole") defined for the sales division 132 "o = sales". In other words, this implies that the user "cn = rob" needs to be added as a member

5   of the sales role 135, while remaining a member of the engineering role 133.

As the scope of the sales role 135 is defined by the subtree 1302, a solution of the prior art is to add an additional entry "cn = rob", having an attribute "nsRoleDN = salerole" as a subentry of the entry 136 "ou = people" of sales

10   division 132. However, this solution is not easy to apply when the sales division and the engineering division are managed by two distinct role administrators, as often happens. Moreover, it allows the user to have only the permissions authorized for the sales role or only the permissions authorized for the engineering role, but not permissions authorized for both. This solution is thus

15   limited.

Another known solution is to create a managed role "cn=ManagedEngRole" possessing the user-oriented entry "cn=bob" and a nested role that contains both the sales role 132 and the role possessing the user "cn=bob". This requires the

20   new nested role entry to be located at the top level of the directory tree structure, which is extremely complicated to administer.

Embodiments of the present invention address the above problem. Embodiments of the present invention implement a new method for extending role scope.

Embodiments in accordance with the present invention propose to extend the

5   scope of role entries based on extra data. In accordance with embodiments of the present invention, the extra data identify an extra scope of the DIT and may be added to role entries to extend their scope. The extra data are enabled in the LDAP metadata.

10   According to an aspect of the invention, the extra data may comprise a special attribute *nsRoleScopeDN* that identifies the extra scope.

For each one of these roles, if the corresponding role entry does not comprise the special attribute *nsRoleScopeDN*, the scope of the role is the scope defined

15   by the *LDAPsubentry* object class, in accordance with the prior art.

On the other hand, if the corresponding role entry comprises the special attribute *nsRoleScopeDN*, the scope of the role entry is the union of the scope defined through the *LDAPsubentry* object class and the extra scope indicated by the

20   special attribute. Thus, a user entry or a group of user entries, who are members of a given role, can have an extra scope that is different from the scope of the given role. Consequently, a user can benefit from a permission restricted to a

particular role, even if he/she is not in the organization for which the role is defined.

Reference is now made to Figure 10, illustrating the different operations

5 performed for checking whether a given user entry E0 is a member of a given

role R1, according to an embodiment of the invention. As in the prior art, the

directory tree comprises role entries. However, in accordance with embodiments

of the invention, some of the role entries may comprise the special attribute

nsRoleScopeDN.

10

At the initial operation 100, the directory server receives the request. Operation

101 retrieves role data associated with role R1. These role data are represented

by a data structure comprising specific information about the role, like the type of

the role, e.g., "nested", the filter condition when the role is filtered and the role

15 distinguished name *dn*. The role data may be stored in a cache to ease the

processing of the request. They are provided from the attributes comprised in the

role definition entry.

Operation 102 determines if entry E0 is in the scope of role R1. This operation

20 may be performed by comparing part of the distinguished names of entry E1 and

role R1.

If entry E0 is in the scope of role R1, operation 103 further checks whether role R1 is nested.

If role R1 is not nested, the server tests, at operation 104, if entry E0 matches the

5   membership condition of role R1

    - If role R1 is filtered, the membership condition corresponds to the filter

    condition as is provided by the role data;

    - If role R1 is managed, the membership condition is related to the role *dn*

    provided by the role data,

10

If entry E0 matches the membership condition of role R1, operation 106

determines that entry E0 is a member of role R1.

If role R1 is nested, at operation 105, the directory server recursively performs

15   operations102 and 104 for each role contained by the nested role. Therefore, the

membership condition that is tested in those steps corresponds to the

membership conditions of the roles contained by the nested role.

If entry E0 is in the role scope of one of the roles contained by role R1 and if

20   entry E0 matches the membership condition of that role, i.e. if entry E0 is a

member of one the role contained by role R1 (test 107), operation 109

determines that entry E0 is a member of role R1.

At operation 102, if entry E0 is not in the scope of role R1, operation 108

determines whether role R1 has the special attribute *nsRoleScopeDN*. If not,

operation 116 determines that entry E0 is not a member of role R1.

5

If role R1 has the special attribute *nsRoleScopeDN*, this special attribute

identifying an extra scope, operation 110 checks if entry E0 is in the extra scope,

and if so, operation 103, described above, is performed.

10   In accordance with this embodiment of this invention, the special attribute

nsRoleScopeDN designates the Distinguished Name of the extra scope.

According to an alternative embodiment of this invention, in addition to the extra

scope, the extra data of a given role may identify an appended role being out of

15   the scope of the role.

The extra scope is thus a scope of the tree structure that includes the scope of

the role being appended. The extra scope may be, for example, the scope of the

appended role. In particular, the extra data may comprise the special attribute

20   identifying the extra scope and another attribute identifying the appended role.

According to this embodiment of the invention, the directory server enables an

entry or a set of entries being in an extra scope to be a member of a role, if the

following conditions are fulfilled:

- the entry or set of entries is/are a member of an appended role;

5   - the role has the extra data;

- the appended role and the extra scope are identified by the extra data of

the role.

It is not required that the entry or set of entries belong to the scope of the role

10   entry, in contrast to the prior art.

With reference to Figure 9, e.g., the entry "cn=rob" belonging to the appended

role "cn=engrole" could be a member of the role "cn=salesrole", if the role entry

"cn=salesrole" is added the extra data and if the extra data identify the appended

15   role "cn=engrole" and an extra scope including the scope 1300.

In accordance with this embodiment, the special attribute, called

nsRoleScopeDN is attached to nested roles and designates the DN of a location

in the tree, the extra scope being the subtree of this location.

20

Nested role entries are provided by an embodiment of the present invention to

contain one or more roles of any type, these roles being in the scope of the

nested role entry. The attribute *nsRoleDN* of the nested role defines the

distinguished name *dn* of the contained roles. The structure of nested roles has

been improved to also enable an appended role that is out of the scope of a

nested role entry to be contained by the nested role entry, and, as a result, to

5   enable a group of entries out of the scope of the nested role to be a member of

.the nested role. To perform such an extension, the value of the *nsRoleScopeDN*

attribute designates the location of an extra scope that includes the appended

role scope, and a value of the *nsRoleDN* attribute identifies the appended role.

10   Consequently, the directory server may use the *nsRoleDN* attribute designating

the DN of an appended role and the *nsRoleScopeDN* attribute designating an

extra scope to enable an entry possessing the designated appended role to

further possess the nested role.

15   The table of Figure 11 illustrates the structure of a nested role having an

extending scope. A nested role entry according to the prior art is identified by the

attribute *dn*, and belongs to the object classes *LDAPsubentry, nsRoleDefinition,*

nsComplexRoleDefinition and *nsNestedRoleDefinition*. Moreover, a role

contained by the nested role is identified by the attribute *nsRoleDn*. According to

20   the prior art, the values of this attribute designate the distinguished names (dn) of

the contained roles. The syntax of these distinguished names may directly give

information about the scope of the contained roles. According to an aspect of the

invention, the distinguished names of the contained roles may indicate that their

scope is different from the scope of the nested role.

The scope extension of nested roles is made possible by the *nsRoleScopeDn*

5   attribute. This attribute is defined in LDAP schema, and may be associated with

the *nsNesledRoleDefinition* object class. Consequently, any nested role entry of

the directory tree may have added to it the *nsRoleScopeDn* attribute. The syntax

of this attribute is a distinguished name syntax.

10   Additionally, the role data associated with nested roles may comprise the extra

scope or extended scope designated by *nsRoleScopeDN*. The extended scope

may be updated in the role data, in response to the *nsRoleScopeDN* attribute

being modified, added or deleted.

15   Figure 12 is a flowchart showing alternatives to operations 108 and 110 of Figure

10, according to an embodiment in accordance with the present invention.

If operation 102 of Figure 10 determines that entry E0 is not in the scope of role

R1, operation 1080 of Figure 12 checks whether role R1 is nested. If role R1 is

20   nested, operation 1081 determines if role R1 further has an extended scope

based on the role data and if so, operation 1100 determines whether entry E0 is

in the extended scope. In response to entry E0 being in the extending scope,

operation 1101 determine whether entry E0 is a member of an appended role

contained by the nested role. This operation may be performed by determining

whether entry E0 is a member of one of the roles contained by nested role R1,

according to operations 105 and 107 of Figure 10. If test 1101 succeeds,

5   operation 1090 determines that entry E0 is a member of role R1. If one of the

previous tests fails, operations 1160 determines that entry E0 is not a member of

role R1.

Alternatively, it is possible that several appended roles may be contained by a

10   nested role, provided that all those appended roles are in the extra scope

An important function related to roles is to enumerate all the roles possessed by

a user. The concept of extending scope for roles involves modifications to that

function. An embodiment in accordance with the present invention based on

15   nested roles makes it possible to use parts of the existing operations used for

enumerating the roles possessed by a user.

Figure 13a is a flowchart representing the different operations implemented to

perform such a function according to the prior art. Figure 13b is a flowchart

20   representing different operations implemented to perform such a function

according to an embodiment of the present invention.

In the prior art, when a request is made to determine the roles possessed by a given user entry, the set of candidate roles is determined from a list of roles associated with the top suffix of the given entry. This list is prepared in advance in a role cache. The role cache is a data structure updated on creating a new role

5   or on deleting an existing role in the subtree of the top suffix. The role cache contains the list of the roles defined in the subtree of the top suffix. Each role of the role cache is also related to the role data.

The existing function for computing *nsrole* is based on testing whether the given

10   entry is a member of a set of candidate roles. For each one of the candidate roles, the directory server tests if the given entry E0 is a member of that role.

Figure 13a illustrates the processing of a request for listing the roles possessed by a given user entry E0

15   - at operation 200, the directory server receives the request for listing the roles possessed by a given user entry E0. In response to the request for enumerating all the roles possessed by a given user entry E0, in operation 200, the directory server proceeds to the computing of *nsrole* attribute for that entry. The *nsrole* attribute is a multi-valued attribute indicating all the roles possessed by a user

20   entry.   (

- the computing of *nsrole* attribute starts with operation 202, that performs access to the top suffix of entry E0;

- operation 204 retrieves the role cache associated with the top suffix of E;

- for each role of the cache role,

- operation 206 retrieves the role data of the current role;

- operation 208 tests if entry E0 is a member of the current role, and if so adds

5   the role to the result and select the next role of the list;

- when all the candidate roles have been tested (operation 210 fails), the

directory server assigns the result to *nsrole*, at operation 212.

Figure 13b will be discussed below, after a discussion of Figure 14. According to

10   another aspect of the present invention, the directory tree structure may

comprise extended roles i.e., roles having extended scope. Consequently, a

given entry E0 may be a member either of one of the roles associated with its top

suffix, or of one of the roles of another top suffix.

15   Figure 14 illustrates parts of a directory tree structure, comprising an extended

role. Table VI comprises the definition of this structure in LDIF, according to an

embodiment of the present invention.

TABLE VI

suffix definition 1
dn : o=suffix_1
objectclass : organization
objectclass : top

organization entry
dn : o=qa,o=suffix_1
objectclass : organization
objectclass : top

user organization unit entry
dn : ou=users,o=qa,o=suffix_1
objectclass : organizationalUnit
objectclass : top

extended role entry
dn : cn=everybody_cross2,o=qa,o=suffix_1
objectclass : LDAPsubentry
objectclass : nsRoleDefinition
objectclass : nsComplexRoleDefinition
objectclass : nsNestedRoleDefinition
nsRoleScopeDN : o=suffix_2
nsroledn : cn=marketing,o=qa,o=suffix_2
nsroledn : cn=staff,o=qa,o=suffix_I

staff role
dn : cn=staff,o=qa,o=suffix_1
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsSimpleRoleDefinition
objectclass: nsManagedRoleDefinition

suffix definition 2
dn : o=suffix_2
objectclass : organization
objectclass : top

organization entry
dn : o=qa,o=suffix_2

objectclass : organization
objectclass : top
user organization unit entry dn : ou=users,o=qa,o=suffix_2 objectclass : organizationalUnit objectclass : top
marketing role dn : cn=marketing,o=qa,o=suffix_2 objectclass : LDAPsubentry objectclass : nsRoleDefinition objectclass: nsSimpleRoleDefinition objectclass: nsManagedRoleDefinition
user entry dn : cn=bob,ou=users,o=qa,o=suffix_2 objectclass : top objectclass : person sn : kap userpassword: secret nsroledn : cn=marketing,o=qa,o=suffix_2
© 2002, SUN Microsystems™

In the prior art, to find all the roles possessed by the given entry 246 "cn=bob,o=user,o=qa,o=suffix2", the directory server tests the candidates roles of its top suffix "o=suffix2". This test indicates that entry 246 is a member of the

5   role 243, "cn=marketing". But, the entry 246 is also a member of an extended role 142 "cn=everybody_cross2" under another top suffix 140 "o=suffix1", which is not found by the function of the prior art.

CONFIDENTIAL

Embodiments of the invention propose to modify the selection of the candidate roles in order to further find the extended roles possessed by a given entry.

5   For each top suffix of the directory tree structure, for each role of the role cache associated with that suffix, the directory server may test if entry E0 is a member of that role according to the operations of the Figure 10 or 12. However, as a directory tree structure may comprise a great number of role entries, this may not be optimum.

10

Alternatively, the following operations may be performed :

a1) executing the function for listing all the role possessed by entry E0, according to the prior art, as illustrated in Figure 13a. This operation provides the roles possessed by entry E0, among the roles associated with its top suffix, these

15   roles not being extended;

b1) for each top suffix of the directory tree structure, testing if entry E0 is a member of one of the extended roles among the candidates roles of that suffix, according to the flowchart of Figures 10 and 12.

20   An embodiment of operation b1) is illustrated by the flowchart of Figure 13b, in accordance with an embodiment of the present invention. To determine the extended roles associated with a top suffix of the tree structure (request 500),

operation 500 retrieves the role cache of the current top suffix. In accordance

with an aspect of this embodiment, the role cache further comprises a field

indicating the list of extensions of the top suffix, the extensions representing the

distinguished names of the extended scopes defined in nested roles under the

5   top suffix. This list is prepared in advance from the role data of those nested

roles.

After retrieving the list of extensions (operation 502), operation 503 determines,

for each extension, whether the given entry E0 is in the scope defined by the

10   extension. This may be done by comparing the distinguished name of the entry

with the distinguished name of the extension.

If entry E0 is in the scope defined by the extension, operation 504 gets the

nested roles associated with the current extension. Operation 506 then tests if

15   entry E0 is a member of each one of these roles, according to the operations of

the flowchart represented in Figure 10 or 12. When entry E0 is determined to be

a member of a role, the role is added to the result (operation 507).

After checking all the roles associated with the current extension (operation 508),

20   the directory server recursively repeats operations 503 to 508 for all the

extensions. When all the extensions have been checked (test 509), the directory

server repeats operations 501 to 509 for the next top suffix of the tree structure.

When all the suffixes have been checked (test 510), the directory server assigns

the result to the nsrole attribute of entry E0 (operation 511).

Reference is now made to Figures 15a to 15c, showing an exemplary structure of

5   a role cache. This structure known in the art, has been modified to enable the

extension of role scope. A role cache maintains the following information for a

given top suffix, as shown in Figure 15a:

- the DN of the top suffix in field L1,

- the list of the roles associated with that top suffix in field L2,

10

In accordance with an aspect of this invention, the role cache also provides a

pointer L3 to the list of the top suffix extensions, defined in nested roles under

this given top suffix. This pointer indicates the DN of an extension L31 of the top

suffix and a pointer L32 to the next extension.

15

More specifically, the list of the roles may be represented with binary search

trees, such as AVL trees (named after their inventors Adelson-Velskii-Landis).

The structure of Figure 15a illustrates the structure of an AVL tree. An AVL tree

representing a role comprises a pointer to the root of the AVL tree, in field L2.

20

Each role is represented by an AVL tree. An AVL node of an AVL tree may be

represented by the structure of Figure 15b, comprising a pointer on the left AVL

node L21, a pointer on the right AVL node L23, and AVL data L22 corresponding

to the role data. When the role is nested, the right or the left AVL node

represents the distinguished names of the roles contained by the nested role.

5   Figure 15c illustrates extensions comprising: Extension DN L31 and Next

Extension L32.

Referring to Figure 17, the AVL data represent the role data. In particular, they

comprise the following information :

10

- the distinguished name of the role entry L221,

- the role type L222 (nested /managed/filtered).

If the role is filtered, the AVL data further comprise the filter condition defined by

15   the role filter.

According to another aspect of the invention, the AVL data may comprise the DN

of the extended scope in field L223. This information is related to the value of

nsRoleScopeDN attribute in the role entry and correspond to one of the

20   extension of the suffix.

Figures 16 and 18a to 18c illustrate an example of a role cache of an extended

role, in accordance with embodiments of the present invention. The field L2 of

the role cache of root suffix "o=suffix1" is a pointer on the AVL tree represented

in Figure 16.

5

The AVL node representing the role "cn=everybody _cross,o=suffix1" of field

L22.1, has a pointer 123.1 on the AVL tree a first contained role 122.2 "cn=staff,

o=qa,o=suffix_2".

10   Moreover, the AVL node representing this first contained role, has a pointer

121.2 on the AVL tree of a second contained role L22.3 "cn=staff,

o=qa,o=suffix_1".

Reference is now made to Figures 18a to 18c, representing the AVL data of the

15   roles L22.1, L22.2 and L22.3, in accordance with embodiments of the present

invention. The field L222.1 of Figure 18a indicates that the role L22.1

"cn=everybody_cross, o=suffix 1" is nested. The AVL data of that role also

comprise the field L223 comprising a distinguished name indicating an extended

scope. The extension of the scope of the role L22.1 is performed using the

20   information contained in fields L22.2 and L223.

Figures 18b and 18c represent the AVL data of the roles contained by the role

"everybody _cross, o=suffix1", in accordance with embodiments of the present

invention. The field L221.2 of Figure 18b indicates that the role having the DN

"cn=staff,o=qa,o=suffix2" stored in field L221.2 is managed. The field L221.3 of

5   Figure 18c indicates that the role having the DN "cn=staff,o=qa,o=suffix12"

stored in field L221.2 is also managed.

In accordance with embodiments of the present invention, on attaching the

nsRoleScopeDN to a nested role, the directory server proceeds to the following

10   update operations :

        i1) updating the field L223 "scope extension DN" in the role data

associated with the nested role;

        ii1) updating the field L22 "extensions" L31 in the role cache of the nested

role suffix.

15

In accordance with another embodiment of this invention, it is possible to enable

a single user in a first scope to be a member of a nested role having a second

scope, the first scope and the second scope being different. This function is

performed by the following operations:

20   i2) creating an appended role, preferably a managed role, that only

possesses the single user;

ii2) adding the attribute *nsRoleScopeDN*, invoking the extra scope and the attribute *nsRoleDN*, invoking the *dn* of the appended role to selected nested roles of the tree structure, in order to enable the single user to be a member of those roles.

5

Referring now to Figure 9, many users, not represented, may be a member of the role "cn=engrole". To enable the single user "cn=rob" of the suffix 134 "o=eng" to be a member of the role 135 "cn=salesrole", according to operation i2), a managed role 137 "cn=managedEngRole" has been created, according to an

10   embodiment of the present invention. The corresponding structure is represented in Figure 19. According to operation ii2), the role 135 "cn=salesrole" is added according to an embodiment of the present invention:

- the attribute *nsRoleScopeDN*, invoking the distinguished name of the extra scope;

15   - the attribute *nsRoleDN*, invoking the distinguished name of the role *managedEngRole* "o=eng,dc= France,dc=sun,dc=com".

In the prior art, the enumeration of the entries possessed by a given role is performed by computing the *nsrole* attribute of candidate entries. The operations

20   for computing *nsrole* attribute according to embodiments of the invention, have been described in reference to Figure 13a.

Embodiments in accordance with the invention also encompass software code, especially when made available on any appropriate computer-readable medium. The expression "computer-readable medium" includes a storage medium such as magnetic or optic, as well as a transmission medium such as a digital or

5   analog signal. Such software code may include data and/or metadata.

This invention also encompasses the software code to be added to existing directory server functionalities to perform anyone of the various new functionalities, as described above, which may be used independently of each

10   other.

On another hand, a number of features have been positively described, using absolute language, to help understanding the LDAP example. Each such feature should be considered as exemplary only, and is not intended to restrict the scope

15   of this invention in any way.

The preferred embodiment of the present invention, extending role scope in a distributed computer system, is thus described. While the present invention has been described in particular embodiments, it should be appreciated that the

20   present invention should not be construed as limited by such embodiments, but rather construed according to the below claims.